# Curl – Building RIA Beyond AJAX

*Rich Internet Applications for the Enterprise*

The Web has brought about an unprecedented level of connectivity and has put more data at our fingertips than ever before, transforming how we access information and services. Enterprises everywhere feel impelled to capitalize on this transformation for two main reasons: the increased reach of Web applications and the reduced IT administration costs that result from Web deployment. Just install or update files on a Web server, and an application is ready for use across an enterprise or around the world, without any need to update software on thousands of client machines. Unfortunately, Web browsers are inherently limited as application execution platforms, causing most Web applications to be much less usable than the older applications that they replace. Thus, this revolution has returned us to the days of "dumb terminal" user interfaces with limited interactivity and vastly inefficient use of communication bandwidth.

As long ago as 1995, a team of MIT computer scientists foresaw this situation and began investigating how to build Rich Internet Applications (RIA) — Web-deployed applications with the same rich user interfaces as locally installed applications. The end result of this research was the Curl technology, which has now developed into a full-featured commercial application platform. Until now, the primary marketing focus for Curl has been in Japan, where Curl is currently used by over 200 industrial customers and is the focus of an ecosystem including more than 40 partner companies that develop applications based on the Curl platform.

Outside of the Curl world, as companies began to migrate or adapt their client interfaces for the Web, they realized that while the Internet as a network may be well suited for their needs, the first-generation Web languages and tools were woefully inadequate to deal with anything more than simple text markup and content delivery. Server-side technologies such as application servers and portal platforms were developed to provide a level of dynamic behavior that was impossible with HTML alone, but they did little to address sluggish user interfaces and poor end-user productivity.

Then, in 2004 and 2005, a couple of high-profile Web applications with enriched user interfaces, GMail and Google Maps, appeared. These applications sparked widespread excitement about being able to enjoy the benefits of Web-deployed applications without suffering the usability drawbacks of classical Web applications. The technical approach used to build these new applications was soon named "AJAX" (Asynchronous JavaScript and XML) and a crescendo of enthusiasm for this new type of application quickly followed. Since then, there have been AJAX conferences, AJAX Web sites, and even an OpenAjax Alliance that, at last count, numbered more than 50 corporate members.

## What are the limitations of today's AJAX tools?

It is important to recognize that AJAX is not a product, nor a standard. It is an umbrella term for the process of using a set of technologies including HTML, CSS (Cascading Style Sheets), and JavaScript to build highly interactive Web application front ends. Compared with classic page-based web applications, AJAX introduces a new Web presentation tier model that is different in three important ways, namely:

1. A client-side engine acts as an intermediate between the User Interface (UI) and the server;
2. User activity leads to JavaScript calls to the client-side engine instead of a page request to the server;
3. XML data, rather than HTML pages, are transferred between the server and the client.

Alternatively stated, AJAX solutions contain a client-side engine, consisting of JavaScript functions, which renders the user interface and communicates with the server in XML format.

However, AJAX still uses existing tools such as JavaScript and DHTML as the basic building blocks and that's where some of the challenges lie. The programmer still bears the burden of being clever enough to combine these disparate tools to implement the desired application behavior. Cleverly used, JavaScript has enabled impressive advances in mapping, e-mail clients, stock charting, shopping carts, and similar applications. These are great improvements, but these applications are pretty simple compared with a typical enterprise or desktop application. The next step in application sophistication puts serious pressure on weak points of the JavaScript model, such as run-time performance and robustness for serious software engineering.

This is where Curl comes in. It is a rich framework that transcends these limitations and complements JavaScript as we take the next step toward true transactional applications for the enterprise. While Curl delivers all three beneficial characteristics of the AJAX model, it provides a more powerful language and runtime environment than AJAX, suitable for building the complex applications that large enterprises need, as some of the examples will highlight later in this paper.

## Basic characteristics of Curl

Curl was conceived by a team of MIT scientists during the late 1990s to provide a fully-featured and robust technology beyond HTML and JavaScript, enabling the development of very rich Internet client applications. The original Curl language was designed to be a *content language* with a unified notation for information, style, and behavior. In that sense, Curl precedes today's AJAX concept.

Curl does not require changes in the back-end infrastructure we have today for Web-based enterprise applications, mostly built upon J2EE or Microsoft's .Net. Curl provides a new client-side technology fully complementary to these server-side alternatives. The unique Curl Platform consists of three components:

1. An extensible programming language (the Curl content language) designed to create interactive Web content.
2. A runtime engine (the Curl RTE) that executes Curl programs and renders the resulting content.
3. An IDE (integrated development environment) to construct rich Internet client-side applications.

Strict adherence to standards such as XML for transmission of data and HTTP for communication enables seamless integration with both Java- and .Net-based back ends.

Here is how a Curl-based application works: Upon an initial request from a user, the back-end server will send down the application code and, optionally, its associated data in a compact file that can be an order of magnitude smaller than a comparable HTML-based application. From that point, the full power of the application is literally in the hands of the user; it runs on the user's local machine, not the server. It will request additional data from the server as needed.

The server is freed from much of the processing it typically does because the application offloads all the presentation work and appropriate business logic tasks onto the client. The network is freed from the constant back-and-forth requests and sending down of fresh web pages that characterize the pre-AJAX model. The huge savings in server load and network bandwidth make Curl-based applications enormously scalable. Incidentally, this is precisely the principle of AJAX, where a local agent pulls data in asynchronously from the server for faster client performance and minimizes network round-trips.

The object-oriented power of the Curl language renders applications very extensible. The runtime environment includes more than 4000 APIs that programmers can use out of the box; developers can build their own features on top, with no limits. This approach is in line with the "invocable service" concept at the heart of trends like Service Oriented Architecture and web services. The resultant productivity gains are significant with this "assembly of components" model of application construction.

## Examples of usage in the enterprise

The combination of client-side processing, the content language concept, and Curl's rich set of user-interface APIs makes Curl uniquely suited for enterprise applications such as the following:

**Dashboards for viewing and understanding enterprise data.**

Curl already supports the communication protocols needed to send data sets from the server to the client machine, and the Curl API includes a rich set of charting features as well as 2D and 3D graphics for building lucid, interactive data displays. Since these data displays are created autonomously on the client machine, they can easily include highly responsive "drill down" capabilities to examine data in more detail. Moreover, Curl's "elastic" graphical layout feature means that dashboard displays can adapt naturally to any window size a user prefers. As the user interactively reconfigures the dashboard display, the current configuration (and even a cache of data received from the server) can be saved on the client machine using Curl's client-side persistent data feature, so the user's next session can begin exactly where the last session left off. Finally, Curl's ability to keep working even when disconnected from the network means that a Curl dashboard can continue to be used for exploring data cached on the client machine even while traveling or otherwise out of reach of a network connection.

**Portals and "mash-ups" that integrate data from multiple sources.**
Acting as the user's agent, a Curl-based portal application can reach out to multiple information sources, directly using multiple data streams to synthesize an overall view of the data. In addition to this "mash-up" capability, a Curl application can embed non-Curl content such as subsidiary browser frames. The many features that make Curl-based dashboards more interactive and visually engaging are useful for portal applications also.

**Interactive forms with intelligent data validation.**
Forms-based transactions and data entry are a key part of most enterprise workflows. The smooth integration of high-grade text formatting and active content within the Curl content language framework means Curl-based forms can be visually attractive and effective from a human factors standpoint while at the same time able to apply sophisticated processing to validate and guide user input. It is even easy to build forms that dynamically open out new panels where required by a specific user's situation. Curl-based forms also benefit from occasionally connected computing: a traveling salesman could use a Curl application to enter notes from a customer visit, or prepare his expense report, while flying back home.

**Live documentation and interactive education.**
The Curl content language is ideally suited for creating dynamic, interactive maintenance manuals, training courses, and other educational or informational materials. The content language framework is at its best when combining interactive calculators, diagrams, visualizations, animations, or simulations with richly formatted text that can also link to or integrate with resources on other Web sites as appropriate. And again, Curl's capability to support occasionally connected computing means that a Curl-based manual can still be used even when a network connection would be inconvenient or unavailable

**Key benefits of Curl**

- Improved application usability through richer, more interactive user interfaces.
- "One-Stop Shopping" for client-side user interface implementation

  Text description and layout design (like HTML)

  Scripting language (like JavaScript)

  Object-oriented programming language (like Java)

  2D/3D graphics and multimedia support (like Flash/Shockwave)
- IT cost savings from Web deployment of Rich Internet Applications.
- Reduced server load and network bandwidth requirements.
- Simpler, faster application development.
- Reduced application maintenance costs.

**Summary**

In today's fast-moving world, Web application developers must respond flexibly to frequent change requests and new requirements. Development managers need to keep development costs low while ensuring an adequate level of technical expertise and application support capability. IT managers are asked to support ever more demanding application features within a budget for server and networking infrastructure.

Meanwhile, users are not fully satisfied with the current levels of Web application capability and quality. Users accustomed to the user interface of client-server applications tend to be frustrated with the usability and functionality of current Web applications. In addition, along with Web access to applications, highly creative users demand sophisticated visualizations and application features, not to mention attractive document presentation and standardized data entry.

Curl is the one solution that can reduce development, maintenance, and infrastructure costs while simultaneously meeting expanded user requirements. In summary, the unique features of the Curl platform combine to provide these benefits for Rich Internet Applications for the enterprise.